LAPORAN PRAKTIKUM PEKAN 7

ALGORITMA DAN PEMROGRAMAN

PENERAPAN KONSEP DASAR PEMROGRAMAN BERORIENTASI OBJEK (OOP) DAN VARIASI METODE (*GETTER*, *SETTER*, FUNGSI) DALAM IMPLEMENTASI STUDI KASUS MAHASISWA DAN MANIPULASI STRING PADA JAVA

Disusun oleh:

Thaariq Salam

2511532022

Dosen Pengampu: Dr. Wahyudi S.T.M.T

Asisten Praktikum: Rahmad Dwirizki Olders

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

TAHUN 2025

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT atas limpahan rahmat dan karunia-Nya, sehingga laporan praktikum mata kuliah Algoritma dan Pemrograman Pekan 7 dengan judul "Penerapan Konsep Dasar Pemrograman Berorientasi Objek (OOP) dan Variasi Metode" dapat terselesaikan dengan baik. Laporan ini disusun sebagai salah satu bentuk tugas praktikum Algoritma dan Pemrograman. Pembahasan utama meliputi konsep *Class* dan Object, penerapan *Encapsulation* melalui metode *Getter* dan *Setter*, perancangan fungsi untuk pengecekan bilangan prima, serta berbagai metode manipulasi tipe data String dalam bahasa Java.

Penyusun menyadari bahwa tersusunnya laporan ini tidak lepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, ucapan terima kasih kami sampaikan kepada:

1. Bapak Dr. Wahyudi, S.T., M.T., selaku Dosen Pengampu mata kuliah Algoritma dan Pemrograman.
2. Saudara Rahmad Dwirizki Olders, selaku Asisten Praktikum, atas bimbingan dan arahannya selama pelaksanaan praktikum.

Akhir kata, penulis berharap laporan ini dapat memberikan manfaat, baik bagi penulis sendiri maupun bagi para pembaca, khususnya dalam memperdalam pemahaman mengenai Pemrograman Berorientasi Objek dan pemanfaatan metode dalam bahasa Java

Padang, 13 November 2025

Penulis

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi yang pesat menuntut penguasaan konsep Pemrograman Berorientasi Objek (OOP), khususnya dalam bahasa Java, yang sangat luas penggunaannya. Dalam OOP, konsep dasar seperti *Class* dan Object menjadi fondasi[1]. Penguasaan metode juga menjadi krusial, baik metode yang berfungsi sebagai *Mutator*(*Setter*) dan *Accessor*(*Getter*) untuk mengelola data objek, maupun perancangan fungsi independen untuk menyelesaikan masalah spesifik[1] seperti pengecekan bilangan prima.

Selain itu, manipulasi data String adalah kemampuan yang wajib dikuasai, karena data teks merupakan bagian tak terpisahkan dari aplikasi modern. Praktikum ini bertujuan untuk mengimplementasikan dan menganalisis konsep tersebut secara komprehensif.

1.2 Tujuan

1. Memahami konsep *Class, Object,* dan prinsip *Encapsulation* (*Getter* dan *Setter*) pada studi kasus mahasiswa.
2. Mampu merancang dan menggunakan fungsi atau metode statis yang mengembalikan nilai (*return value*) untuk menyelesaikan masalah logis (studi kasus bilangan prima).
3. Memahami dan mengimplementasikan berbagai metode manipulasi *String* seperti *length(), concat(), toUpperCase(), startsWith(),* dan *contains().*

1.3 Manfaat

Melalui praktikum ini, mahasiswa diharapkan memperoleh pemahaman mendalam mengenai fondasi pemrograman berorientasi objek, kemampuan mendefinisikan dan memanggil metode, serta keterampilan dalam mengolah data teks menggunakan fitur-fitur yang disediakan oleh tipe data *String* dalam Java..

BAB II

PEMBAHASAN

2.1 Pratikum Konsep *Class* dan *Object (Encapsulation)*

Praktikum ini berfokus pada perancangan sebuah *Class* sebagai rancangan data mahasiswa dan implementasi *encapsulation* untuk mengelola data tersebut. *Class* adalah sebuah *blueprint* atau cetak biru yang mendefinisikan objek [1]

2.1.1 Kode program

Kode program pada gambar 2.1 menunjukkan struktur *Class* dengan variabel global *private*, dan kode pada gambar 2.2 adalah program pemanggilnya.

```
1  package pekan7_2511532022;
2
3  public class Mahasiswa_2511532022 {
4      // variabel global
5      private int nim;
6      private String nama,nim2;
7      // membuat mutator (setter)
8      public void setNim (int nim) {
9          this.nim=nim;
10     }
11     public void setNim2 (String nim2) {
12         this.nim2=nim2;
13     }
14     public void setNama (String nama) {
15         this.nama=nama;
16     }
17
18     // membuat accessor (getter)
19     public int getNim() {
20         return nim;
21     }
22     public String getNim2() {
23         return nim2;
24     }
25     public String getNama() {
26         return nama;
27     }
28
29     // metode lain
30     public void Cetak() {
31         System.out.println("Nim : " + nim);
32         System.out.println("Nama : " + nama);
33     }
34     public void Cetak2() {
35         System.out.println("Nim : " + nim2);
36         System.out.println("Nama : " + nama);
37     }
38 }
```

Gambar 2.1

```java
package pekan7_2511532022;

public class PanggilMahasiswa_2511532022 {
    public static void main (String[]args) {
        Mahasiswa_2511532022 a=  new Mahasiswa_2511532022();
        a.setNim(2022);
        a.setNama("Thaariq Salam");
        System.out.println(a.getNim());
        System.out.println(a.getNama());
        a.Cetak();
    }
}
```

Gambar 2.2

2.1.2 Langkah kerja

1) Buat *class* baru bernama *Mahasiswa_2511532022.java* dan deklarasikan variabel instan (*nim*, *nama*) sebagai *private*.
2) Definisikan metode *Mutator* (*Setter*) seperti *setNim()* dan *setNama()* untuk mengisi nilai ke variabel *private*.
3) Definisikan metode *Accessor* (*Getter*) seperti *getNim()* dan *getNama()* untuk mengambil nilai variabel *private*.
4) Buat *class* pemanggil baru bernama *PanggilMahasiswa_2511532022.java* yang memiliki metode *main*.
5) Di dalam *main*, buat sebuah objek dari *class Mahasiswa_2511532022*.
6) Panggil metode *Setter* pada objek tersebut untuk mengisi data *NIM* dan *Nama* (*a.setNim(2022); a.setNama("Thaariq Salam");*).
7) Cetak data menggunakan metode *Getter* (a.*getNim()*) dan metode *Cetak()* yang tersedia di *class Mahasiswa*..

2.1.3 Analisis hasil

```
2022
Thaariq Salam
Nim : 2022
Nama : Thaariq Salam
```

Gambar 2.3

Dalam program *Mahasiswa_2511532022.java,* variabel instan seperti *nim* dan *nama* dideklarasikan sebagai *private*, yang merupakan inti dari *Encapsulation* untuk melindungi data [1]. Pengaksesan dan modifikasi data dilakukan melalui *Mutator (Setter*) seperti *setNim* dan *Accessor (Getter)* seperti *getNama*[1].

Pada program *PanggilMahasiswa_2511532022.java*, sebuah objek (*a*) dibuat dari *Mahasiswa_2511532022* dan nilai diatur menggunakan *Setter*, kemudian diambil menggunakan *Getter* dan metode *Cetak()*..

2.2 Praktikum Manipulasi String Dasar dan Konkatenasi

Praktikum ini mempelajari metode-metode bawaan yang disediakan oleh kelas *String* untuk memproses data teks.

2.2.1 Kode program

Kode program 2.4 dan 2.5 mengilustrasikan berbagai metode String dan perbedaan dalam operator penambahan (+).

```
1  package pekan7_2511532022;
2
3  public class String1_2511532022 {
4      public static void main (String[]args) {
5          String salam= "Assalamualaikum";
6          System.out.println("panjang salam adalah: " + salam.length());
7          System.out.println(salam.toUpperCase());
8          System.out.println(salam.toLowerCase());
9          System.out.println(salam.indexOf("salam"));
10     }
11 }
```

Gambar 2.4

```java
package pekan7_2511532022;

public class String2_2511532022 {
    public static void main (String[]args) {
        String firstName = "Syifa";
        String lastName = "Muhassanah";
        String txt1 = "Dosen\"intelektual\" kampus";
        System.out.println("Nama lengkap: " + firstName + " " + lastName);
        System.out.println("Nama lengkap: " + firstName.concat(" ").concat(lastName));
        System.out.println(txt1);
        int x= 10;
        int y= 20;
        int z= x+y;
        System.out.println("x + y= " + z);
        String a= "10";
        String b= "20";
        String c= a+b;
        System.out.println("a + b= " + c);
        String v= a+y;
        System.out.println("a + y= " + v);
    }
}
```

Gambar 2.5

2.2.2 Langkah kerja

1. Buat *class* String1_2511532022.java dan deklarasikan variabel *String* (*salam*).
2. Implementasikan dan cetak hasil dari berbagai metode dasar *String: length(), toUpperCase(), toLowerCase(), dan indexOf().*
3. Buat *class* String2_2511532022.java dan deklarasikan beberapa variabel *String* dan *int*.
4. Konkatenasi *String* menggunakan operator + dan metode *concat().*
5. Demonstrasikan perbedaan antara penjumlahan numerik (*int x + int y*) dengan konkatenasi *String (String a + String b* dan *String a + int y*).

2.2.3 Analisis hasil

```
panjang salam adalah: 15
ASSALAMUALAIKUM
assalamualaikum
2
```

Gambar 2.6

```
Nama lengkap: Syifa Muhassanah
Nama lengkap: Syifa Muhassanah
Dosen"intelektual" kampus
x + y= 30
a + b= 1020
a + y= 1020
```

Gambar 2.7

- Gambar 2.6 menunjukkan penggunaan *salam.length()* untuk mengetahui panjang string, *toUpperCase()* dan *toLowerCase()* untuk mengubah huruf, serta *indexOf (salam)* untuk mencari posisi *substring*.
- Gambar 2.7 menunjukkan dua cara untuk menggabungkan string: menggunakan operator + dan menggunakan metode `concat()`.

Terdapat perbedaan fundamental pada operator + [1]:

- Jika digunakan pada dua variabel *int* (*x + y*), hasilnya adalah penjumlahan numerik (30).
- Jika digunakan pada dua variabel *String* (*a + b*), hasilnya adalah konkatenasi teks (1020).
- Jika digunakan pada *String* dan *int* (*a + y*), Java akan secara otomatis menganggap y sebagai *String*, sehingga hasilnya adalah konkatenasi (*1020*).

2.3 Praktikum Pengecekan Bilangan Prima

Praktikum ini mencoba membuat sebuah fungsi yang bertujuan spesifik untuk menentukan apakah sebuah bilangan yang dimasukkan pengguna merupakan bilangan prima atau bukan.

2.3.1 Kode program

Gambar 2.8 menunjukkan implementasi fungsi dengan nilai kembalian bertipe *boolean* dan pemrosesan *Input* dari *user*

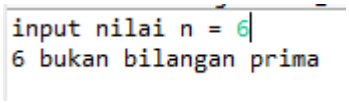
```java
package pekan7_2511532022;

import java.util.Scanner;

public class BilanganPrima_2511532022 {
    public static boolean isPrime (int n) {
        int factors = 0;
        for (int i = 1; i<= n; i++){
            if (n % i == 0) {
                factors++;
            }
        }
        return (factors == 2);
    }
    public static void main(String[]args) {
        Scanner input = new Scanner(System.in);
        System.out.print("input nilai n = ");
        int a = input.nextInt();
        if (isPrime(a)) {
            System.out.println(a + " bilangan prima");
        } else {
            System.out.print(a + " bukan bilangan prima");
        }
    }

}
```

Gambar 2.8

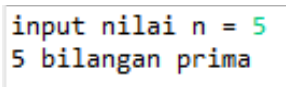
2.3.2 Langkah kerja

1. Buat *class* baru bernama *BilanganPrima_2511532022.java.*
2. Rancang sebuah fungsi statis dengan *return type boolean* bernama *isPrime*(*int n*).
3. Di dalam fungsi *isPrime*, gunakan perulangan (*for*) untuk menghitung jumlah faktor pembagi (*factors*) dari angka *n*.
4. Fungsi harus mengembalikan (*return*) nilai *true* jika *factors* sama dengan 2, dan *false* jika tidak.
5. Di dalam metode *main*, gunakan *Scanner* untuk meminta *input* nilai *n* dari pengguna.
6. Panggil fungsi *isPrime*(*n*) di dalam struktur kondisional *if-else* untuk menentukan apakah *input* merupakan bilangan prima.

2.3.3 Analisis hasil



Gambar 2.9



Gambar 2.10

Metode *public static boolean isPrime* (*int n*) adalah sebuah fungsi yang mengembalikan nilai *boolean* (*true atau false*). Logika di dalamnya menggunakan perulangan (*for*) untuk menghitung jumlah faktor pembagi dari *n*. Suatu bilangan dianggap prima jika jumlah faktornya tepat dua (*factors == 2*), yang berarti bilangan tersebut hanya habis dibagi satu dan dirinya sendiri. Hasil kembalian fungsi ini kemudian digunakan dalam blok kondisional (*if-else*) di metode *main* untuk mencetak *output* kepada pengguna..

2.4 Praktikum Integrasi OOP, Input, dan Validasi String

Praktikum ini menggabungkan penggunaan objek, *input* dari *user* menggunakan *Scanner*, dan pemanfaatan metode *String* untuk melakukan validasi sederhana terhadap data NIM.

2.4.1 Kode program

Kode program 2.3 ini menunjukkan bagaimana cara membuat nilai yang tidak bisa diubah dengan *keyword* final untuk menghitung keliling lingkaran. Contoh konstanta dan perhitungan

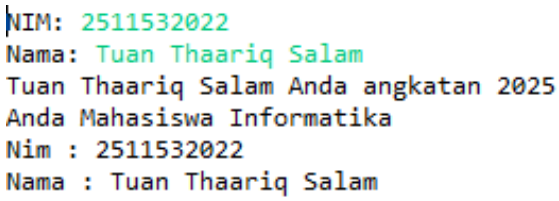
```java
package pekan7_2511532022;

import java.util.Scanner;

public class PanggilMahasiswa2_251153022 {
    public static void main (String[]args) {
        Scanner input = new Scanner(System.in);
        System.out.print("NIM: ");
        String x= input.nextLine();
        System.out.print("Nama: ");
        String y = input.nextLine();
        Mahasiswa_2511532022 a=  new Mahasiswa_2511532022();
        a.setNim2(x);
        a.setNama(y);
        if(x.startsWith("25")) {
            System.out.println(y + " Anda angkatan 2025");
        }
        if(x.contains("1153")) {
            System.out.println("Anda Mahasiswa Informatika");
        }
        a.Cetak2();
        input.close();
    }
}
```

Gambar 2.11

2.4.2 Langkah kerja

1) Buat *class PanggilMahasiswa2_251153022.java* dan *java.util.Scanner*.
2) Di dalam metode *main*, buat objek *Scanner* dan minta *input* NIM (sebagai *String x*) dan Nama (sebagai *String y*) dari pengguna.
3) Buat *object Mahasiswa_2511532022* (*a*) dan gunakan *Setter* yang menerima *String* (*setNim2(x)*) untuk menyimpan data *input*.
4) Gunakan metode *String startsWith("25")* untuk melakukan pengecekan angkatan, lalu cetak hasilnya.
5) Gunakan metode *String contains("1153")* untuk melakukan pengecekan kode program studi, lalu cetak hasilnya.
6) Cetak data objek menggunakan *a.Cetak2()* dan tutup objek *Scanner*.

2.4.3 Analisis hasil



Gambar 2.12

Program ini mengambil *input* NIM dan Nama dari pengguna menggunakan *Scanner*. Meskipun NIM disimpan sebagai *String* (*x*), ia tetap dapat digunakan untuk menginisialisasi objek Mahasiswa melalui *a.setNim2*(x). Validasi dilakukan menggunakan dua metode *String*:

- *x.startsWith("25")*: Digunakan untuk mengecek apakah NIM dimulai dengan kode angkatan tertentu (misalnya 2025).
- *x.contains("1153")*: Digunakan untuk mengecek apakah NIM mengandung kode program studi tertentu (misalnya 1153 untuk Informatika).

Penggunaan metode *String* ini memungkinkan program untuk menganalisis dan memberikan *feedback* kepada pengguna berdasarkan pola data yang dimasukkan, menunjukkan fleksibilitas tipe data *String* dalam pengolahan data..

BAB III

PENUTUP

3.1 Kesimpulan

Dari praktikum ini dapat disimpulkan bahwa pemrograman berorientasi objek (OOP) melalui konsep *class* dan *object* merupakan cara yang efektif untuk memodelkan entitas dunia nyata [1]. Prinsip *encapsulation* diamankan dengan metode *getter* dan *setter* untuk menjaga integritas data [1]. Penguasaan perancangan metode/fungsi adalah hal penting untuk membuat kode yang modular, ditunjukkan oleh fungsi statis *isPrime* yang mengembalikan nilai *boolean*.

Terakhir, tipe data *String* memiliki serangkaian metode bawaan (*length()*, *concat()*, *startsWith()*, *contains()*) yang sangat esensial untuk memproses dan memvalidasi data teks dalam aplikasi Java. Dengan demikian, tujuan praktikum untuk menguasai konsep OOP, metode, dan manipulasi string telah tercapai..

DAFTAR PUSTAKA

[1] P. Deitel dan H. Deitel, Java How to Program, Early Objects, 11th ed. Boston, MA: Pearson, 2018